

CHAPTER

9

Using Unnamed and Named Pipes

<i>Overview of Pipes</i>	153
<i>Using Unnamed Pipes</i>	154
<i>Unnamed Pipe Syntax</i>	154
<i>Using Redirection Sequences</i>	155
<i>Unnamed Pipe Example</i>	155
<i>Using Named Pipes</i>	155
<i>Named Pipe Syntax</i>	156
<i>Using the CALL RECONNECT Routine</i>	157
<i>Using Named Pipes in SCL</i>	157
<i>Named Pipe Examples</i>	158
<i>Simple Named Pipes: One Client Connected to One Server</i>	158
<i>One Server Connected to Several Clients</i>	159
<i>The NOBLOCK Option</i>	161
<i>The CALL RECONNECT Routine</i>	163

Overview of Pipes

A *pipe* is a channel of communication between two processes. For example, with the SAS System and OS/2, you can use a specialized OS/2 application to provide information to your SAS session or vice versa.

Pipes can be one-way or two-way. With a one-way pipe, one application writes data to the pipe, and the other application reads from it. With a two-way pipe, both applications can read and write data. Pipes can be either unnamed or named:

unnamed pipe

Also called an anonymous pipe (or simply a pipe), this type of pipe is always one way. It is typically used to communicate between a parent process and a child process. Within SAS, the SAS System is the parent process that invokes (and reads data from) a child process.

named pipe

This type of pipe can handle one-way or two-way communication between two unrelated processes. That is, one process is not started by the other. In fact, it is possible to have two applications communicate over a pipe on a network. You can use named pipes within SAS to communicate with other applications or even with another SAS session.

Using Unnamed Pipes

Unnamed pipes enable you to run a program outside the SAS System and redirect the program's input, output, and error messages to the SAS System. This capability enables you to capture data from a program that is external to the SAS System without creating an intermediate data file.

For unnamed pipes to work with OS/2 applications that are external to the SAS System, the application program must read data from standard input (STDIN), write output to standard output (STDOUT), and write errors to standard error (STDERR). These files have numeric file handles associated with them, as follows:

File	File Handle
STDIN	0
STDOUT	1
STDERR	2

When the SAS System captures STDERR from another application, the error messages are routed by default to the SAS log. If you want to write to STDIN in another application, you can use a PUT statement in a SAS DATA step. Because the SAS System can write to STDIN and capture from STDOUT in the same application, unnamed pipes can be used to send data to an external program, as well as to capture the output and error messages of the same program. You can use redirection sequences to redirect STDIN, STDOUT, and STDERR. For more information, see "Using Redirection Sequences" on page 155 or your OS/2 documentation.

Unnamed Pipe Syntax

To use an unnamed pipe, issue a FILENAME statement with the following syntax:

FILENAME *fileref* PIPE '*program-name*' *option-list*

You can use the following arguments with this form of the FILENAME statement:

fileref

is any valid fileref, as described in "Referencing External Files" on page 82.

PIPE

is the device-type keyword that tells the SAS System that you want to use an unnamed pipe.

program-name

specifies the external OS/2 application program. This argument must fully specify the pathname to the program, or the path to the directory containing the program must be contained in the OS/2 PATH environment variable. This argument can also contain program options. For example, you can specify the following argument to indicate that you want to run the STOCKMKT program on all stocks:

```
'stockmkt.exe -all'
```

option-list

can be any of the options that are valid in the FILENAME statement, such as the LRECL= or RECFM= options. For a complete list of options that are available for the FILENAME statement under OS/2, see "FILENAME" on page 292.

Using Redirection Sequences

Any OS/2 application that accommodates standard input, output, and error messages can use the unnamed pipe feature. Because many OS/2 system commands use standard input, output, and error messages, you can use these commands with unnamed pipes within SAS. Unless you specify otherwise, an unnamed pipe directs STDOUT and STDERR to two different files. To combine the STDOUT and STDERR into the same file, use redirection sequences. The following is an example that redirects STDERR to STDOUT for the OS/2 DIR command:

```
filename listing pipe 'dir *.sas 2>&1';
```

In this example, if any errors occur in executing this command, STDERR (2) is redirected to the same file as STDOUT (1). This is an example of the SAS System's ability to capitalize on operating environment capabilities. This feature of redirecting file handles is a function of the OS/2 operating environment rather than of the SAS System.

Unnamed Pipe Example

In the following example, you use unnamed pipes to produce financial reports. This example assumes that you have a stand-alone program that updates stock market information from a financial news bureau. You need the SAS System to run a stock market report with the most recently created data from the stock market program. Here is how you create and use the pipe within your SAS session:

```
filename stocks pipe 'stockmkt.exe -all';
data report;
  infile stocks;
  input stock $ open close change;
run;
proc print;
  var stock open close change;
  sum change;
  title 'Stock Market Report';
run;
```

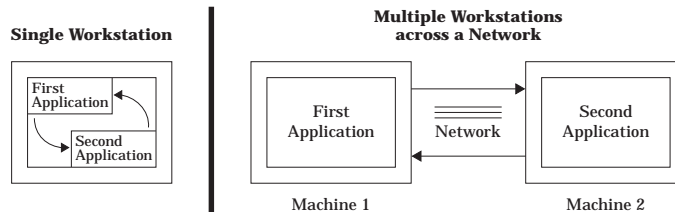
In this example, the PIPE device-type keyword in the FILENAME statement indicates that the fileref STOCKS is an unnamed pipe. The STOCKMKT.EXE reference is the name of the stand-alone program that generates the stock market data. The INFILE statement causes the SAS System to execute the STOCKMKT.EXE program and read the data in the pipe from it. The STOCKMKT.EXE program is executed without your being aware that it is running. Because the fileref STOCKS has already been defined as an unnamed pipe, the standard output from STOCKMKT.EXE is redirected to the SAS System and captured through the INFILE statement. The SAS program reads in the variables and uses the PRINT procedure to generate a printed report. Any error messages that are generated by STOCKMKT.EXE appear in the SAS log.

Using Named Pipes

The *named pipes* capability is one of the most powerful tools available in the SAS System under OS/2 for communicating with other applications. The named pipes

feature enables bidirectional data or message exchange between applications that are on the same machine or applications that are on separate machines across a network. Figure 9.1 on page 156 illustrates these two basic methods of communication.

Figure 9.1 Communication Using Named Pipes



The applications can be SAS sessions or other OS/2 applications. For example, using a named pipe you can use the PRINTTO procedure to direct the results from SAS procedures to another OS/2 application. Therefore, you can have either multiple SAS sessions that communicate with each other or one SAS session that communicates with another OS/2 application.

Whether you are communicating between multiple SAS sessions or between a SAS session and another OS/2 application that supports named pipes, the pipes are defined in a client/server relationship. One process is defined as the *server*, while one or more other processes are defined as *clients*. In this configuration, you can have multiple clients send data to the server or the server send data to the various clients. Named pipes enable you to coordinate processing between the server and clients by using various options.

Named Pipe Syntax

You can use a named pipe anywhere that you use a fileref in the SAS System. To use a named pipe, issue a FILENAME statement with the following syntax:

FILENAME *fileref* NAMEPIPE '*pipe-specification*' <*named-pipe-options*>;

You can use the following arguments with this form of the FILENAME statement:

fileref

is any valid fileref as described in “Referencing External Files” on page 82.

NAMEPIPE

is the device-type keyword that tells the SAS System that you want to use a named pipe.

pipe-specification

is the name of the pipe.

This argument has two mutually exclusive forms:

\\.\PIPE*pipe-name*

indicates that you are establishing a pipe on a single PC or defining a server pipe across a network. The *pipe-name* argument specifies the name of the pipe.

server-name\PIPE*pipe-name*

indicates that you are establishing a client pipe over a network named-pipe server. Remember to include the double backslash (\\) in this situation. The *pipe-name* argument specifies the name of the client pipe. The *server-name* argument specifies the name of the named-pipe server.

named-pipe-options

can be any of the following. The default value is listed first:

SERVER | CLIENT

indicates the mode of the pipe. **SERVER** is the default.

BLOCK | NOBLOCK

indicates whether the client or server is to wait for data to be read if no data are currently available. **BLOCK** indicates to wait, and it is the default.

NOBLOCK indicates not to wait; control is returned immediately to the program if no data are available in the pipe. Writing to the pipe always implies **BLOCK**.

BYTE | MESSAGE

indicates the type of pipe. **BYTE** is the default. The difference between a **BYTE** pipe and a **MESSAGE** pipe is that a **MESSAGE** pipe includes an encoded record length, whereas a **BYTE** pipe does not.

RETRY=*seconds*

indicates the amount of time that the client or server should wait to establish the pipe. The minimum value for *seconds* is 10. This is also the default. This option allows time for synchronization of the client and server.

There are two values for the *seconds* argument that indicate special cases:

- 2 indicates the client should wait the amount of time that is defined by the server's **RETRY=** option. If this option is used, the **SERVER** option must always be active or the pipe connection fails.
- 1 indicates the client or server should wait indefinitely for the pipe connection.

EOFCONNECT

is valid only when you are defining the server, and it indicates that if an end-of-file (EOF) character is received from a client, the server should try to connect to the next client.

All of these options are consistent with the terminology that is used in OS/2 programmers' reference guides.

Using the CALL RECONNECT Routine

There is a special SAS CALL routine that works with named pipes. The **CALL RECONNECT** routine enables the server to disconnect the current client and try to connect to the next available client. Normally, the pipe is terminated when the client side of the pipe sends an end-of-file character to the server. To break the pipe connection at any time, the server SAS session can issue a **CALL RECONNECT** statement. For an example of this routine, see "The **CALL RECONNECT** Routine" on page 163.

Using Named Pipes in SCL

To establish named pipes by using SCL code, you must use the **FOPEN** function to open a file (or pipe) before you can access it. In doing so, you must specify the appropriate open mode for both the client and server applications so that the two can communicate over the pipe. Here is a summary of the different modes that you can use:

If the server accesses the pipe as...	then the client must access the pipe as...
I (input)	O (output)
O (output)	S (sequential)
U (update)	O (output) or S (sequential)

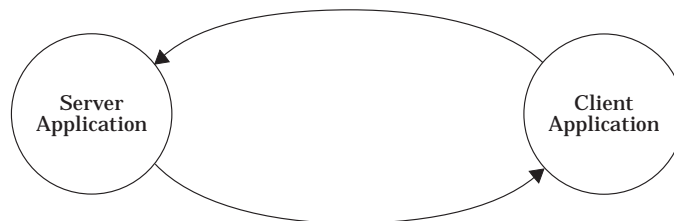
Named Pipe Examples

The best way to understand named pipes is to examine several different examples that illustrate their use. In most of the examples in this section, the named pipe is established between two SAS sessions. However, named pipes work between the SAS System and other applications that support named pipes.

Simple Named Pipes: One Client Connected to One Server

The simplest named pipe configuration is one server connected to one client, as shown in Figure 9.2 on page 158.

Figure 9.2 One Server Connected to One Client



In the following example, a named pipe that is called WOMEN is established between two SAS sessions. The server SAS session selectively sends data to the client SAS session. You can start the server or the client first; one waits 30 seconds for the other to connect.

In the first SAS session, create a named pipe as a server:

```

/* Creates a pipe called WOMEN, acting */
/* as a server. The server waits 30    */
/* seconds for a client to connect.    */
filename women namepipe '\\.\pipe\women'
        server retry=30;
/* This code writes three records into */
/* the named pipe called WOMEN.        */
data class;
  input name $ sex $ age;
  file women;
  if upcase(sex)='F' then
    put name age;
  cards;
MOORE M 15
JOHNSON F 16
DALY F 14
ROBERTS M 14
PARKER F 13
  
```

;

In the second SAS session, you can use SAS statements to exchange data between the two SAS sessions. For example, you can submit the following program from the client session:

```

/* Creates a pipe called WOMEN, acting */
/* as a client. The client waits 30 */
/* seconds for a server to connect. */
filename in namepipe '\\.\pipe\women' client
      retry=30;
data female;
  infile in;
  input name $ age;
proc print;
run;

```

The following program is another example of a single client and server. This example illustrates how to use the PRINTTO procedure to direct results from the SUMMARY procedure to another OS/2 application. The example uses a named pipe called RESULTS:

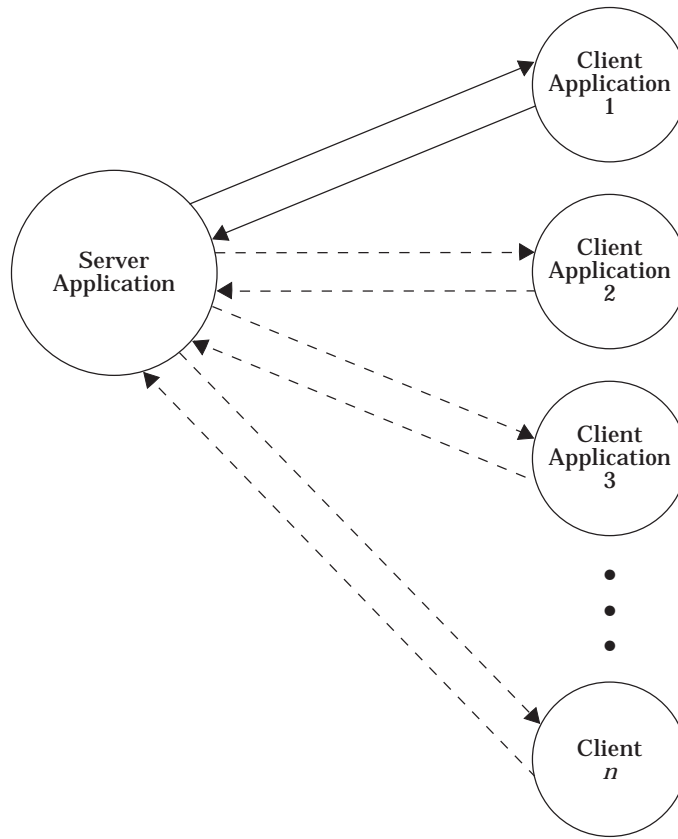
```

filename results namepipe '\\.\pipe\results'
      server retry=60;
proc printto print=results new;
run;
proc summary data=monthly;
run;

```

One Server Connected to Several Clients

You can choose to have one server connected to several clients. In this case, the named pipe configuration looks like that shown in Figure 9.3 on page 160.

Figure 9.3 One Server Connected to Several Clients

In this configuration, the data connection is initially between the server and the first client. When this connection is terminated, the server connects to the second client, and so on. The connection can return to the first client after the last client's connection is broken if your program is set up to do so.

You must use the EOFCONNECT option to cause the connection to move properly from one client to the next. The following is an example of using the EOFCONNECT option with one server SAS session and two clients. The clients can be on the same PC or on a PC that is connected across a network.

In the first SAS session, submit the following statements:

```

/* Creates a pipe called SALES, acting */
/* as a server. The server waits 30 */
/* seconds for a client to connect. */
/* After the client has disconnected, */
/* this server SAS session tries to */
/* connect to the next available client */
filename daily namepipe '\\.\pipe\sales'
server eofconnect retry=30;
/* This program reads in the daily */
/* sales figures sent from each client.*/
data totsals;
infile daily;
input dept $ item $ total;
run;

```

In the second SAS session, submit the following statements:


```

/* Creates a pipe called SALES, acting */
/* as a client. The client waits forever */
/* for a server to connect. After the */
/* first client has disconnected, the */
/* second client connects with the server.*/
/* The first client is the TOYS dept. */
filename dept1 namepipe '\\.\pipe\sales'
      client retry=-1;
data toys;
  input item $ total;
  dept='TOYS';
  file dept1;
  put dept item total;
  cards;
DOLLS 100
MARBLES 10
BLOCKS 50
GAMES 60
CARS 40
;
/* The second client is the SPORTS dept.*/
/* These data could come from a separate */
/* SAS session. */
filename dept2 namepipe '\\.\pipe\sales'
      client retry=-1;
data sports;
  input item $ total;
  dept='SPORTS';
  file dept2;
  put dept item total;
  cards;
BALLS 30
BATS 65
GLOVES 15
RACKETS 75
FISHING 20
TENTS 115
HELMETS 45
;

```

The NOBLOCK Option

The following example uses the NOBLOCK option to specify that the program should continue to execute if no data are available when the pipe is read. If the default value of BLOCK had been used, then the pipe would wait indefinitely until data were found in the pipe. The EOFCONNECT option is also used to tell the server that when a client sends an end-of-file (EOF) character, the server should try to connect with a new client. The RETRY= option tells the server to look for any new clients for 20 seconds while the client waits indefinitely on a server. The clients can be on the same PC or on a PC that is connected across a network. A server connects to one client at a time, and the clients queue in a serial order waiting to connect to the server.

First, submit the following statements in the SAS server session:

```

/* Defines a named pipe called LINE. */
/* Use the NOBLOCK option to specify */

```

```

/* that if no data are available when */
/* the read is performed, then continue.*/
/* Use the EOFCONNECT option to tell */
/* the server to try to connect with a */
/* new client if an end-of-file is */
/* encountered. Use the RETRY= option */
/* to tell the server to look for any */
/* new clients for 20 seconds. */
filename data namepipe '\\.\pipe\line' server
noblock eofconnect retry=20;
/* This DATA step reads in all data */
/* from any clients connected to the */
/* named pipe called LINE. */
data all;
infile data length=len;
input @;
/* If the length of the incoming */
/* record is 0, then no data were */
/* found in the pipe; otherwise, */
/* read the incoming data. */
if len ne 0 then
do;
input machine $ width weight;
output;
end;
run;
proc print;
run;

```

Each of the following client DATA steps can be executed on several different PCs that are connected across a network:

```

/* Defines a named pipe called LINE. */
/* The RETRY= option is set such that */
/* the clients wait forever until a */
/* server is available */
/* (that is, RETRY=-1). */
filename data namepipe '\\.\pipe\line'
client retry=-1;
/* This is information from the */
/* first machine/client. */
data machine1;
file data;
input width weight;
machine='LINE_1';
put machine width weight;
cards;
5.3 18.2
3.2 14.3
4.8 16.9
6.4 20.8
4.3 15.4
6.1 19.5
5.6 18.9
;

```

```

        /* This is information from the */
        /* second machine/client.      */
data machine2;
  file data;
  input width weight;
  machine='LINE_2';
  put machine width weight;
  cards;
4.3 17.2
5.2 18.4
6.8 19.9
3.4 14.5
5.3 18.6
4.1 17.1
6.6 19.5
;

```

The CALL RECONNECT Routine

The following example demonstrates how to set up a named pipe server to establish a connection with two clients. (For this example, you need three active SAS sessions.) In this example, the CALL RECONNECT routine reconnects to the next client on the named pipe if at least 30 seconds have passed since the previous client has sent any data. Each client is a data entry operator that sends data to the server SAS session.

In the server SAS session, submit the following statements:

```

filename data namepipe '\\.\pipe\orders'
                server noblock eofconnect retry=30;
data all;
  infile data length=len missover;
  input @;
  /* If the length of the incoming */
  /* record is 0, then no data were */
  /* found in the pipe; otherwise,  */
  /* read the incoming data         */
  if len ne 0 then
  do;
    input operator $ item $ quantity $;
    if item='' or quantity='' then
      delete;
    else
      output;
    put operator= item= quantity=;
  end;
  /* If no data are being transmitted,*/
  /* try reconnecting to the next     */
  /* available client.                 */
  else
  do;
    /* Use the named pipe fileref */
    /* as the argument of */
    /* CALL RECONNECT. */
    call reconnect('data');
  end;
run;

```

In the second SAS session, which is the first data entry operator, submit the following statements:

```
filename data namepipe '\\.\pipe\orders'
      client retry=-1;
data entry1;
  if _n_=1 then
  do;
    window entry_1
      #1 @2 'ENTER STOP WHEN YOU ARE FINISHED'
      #3 @5 'ITEM NUMBER - ' item $3.
      #5 @5 'QUANTITY - ' quantity $3.;
  end;
  do while (upcase(_cmd_) ne 'STOP');
    display entry_1;
    file data;
    put 'ENTRY_1' +1 item quantity;
    item='';
    quantity='';
  end;
  stop;
run;
```

In the third SAS session, which is the second data entry operator, submit the following statements:

```
filename data namepipe '\\.\pipe\orders'
      client retry=-1;
data entry2;
  if _n_=1 then
  do;
    window entry_2
      #1 @2 'ENTER STOP WHEN YOU ARE FINISHED'
      #3 @5 'ITEM NUMBER - ' item $3.
      #5 @5 'QUANTITY - ' quantity $3.;
  end;
  do while (upcase(_cmd_) ne 'STOP');
    display entry_2;
    file data;
    put 'ENTRY_2' +1 item quantity;
    item='';
    quantity='';
  end;
  stop;
run;
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OS/2® Environment, Version 8*, Cary, NC: SAS Institute Inc., 1999. 448 pp.

SAS® Companion for the OS/2® Environment, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-521-3

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.® indicates USA registration.

IBM® and OS/2® are registered trademarks or trademarks of International Business Machines Corporation. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.